Distributed Security Controls in Industrial Control Systems

Adam Spanier
School of Interdisciplinary Informatics
University of Nebraska at Omaha
Omaha, NE USA
aspanier@unomaha.edu

Kendra Herrmann
School of Interdisciplinary Informatics
University of Nebraska at Omaha
Omaha, NE USA
kherrmann@unomaha.edu

Perry Donahue
School of Interdisciplinary Informatics
University of Nebraska at Omaha
Omaha, NE USA
pdonahue@unomaha.edu

Matthew Popelka
School of Interdisciplinary Informatics
University of Nebraska at Omaha
Omaha, NE USA
mpopelka@unomaha.edu

Brevin Wagner
School of Interdisciplinary Informatics
University of Nebraska at Omaha
Omaha, NE USA
bwagner@unomaha.edu

Abstract—**The use of decentralized security controls (DSC) like blockchains or fingerprinting mechanisms is not new in ICS security operations. The majority of research aimed at better understanding DSC implementations in ICS networks focuses primarily on the effectiveness of a single DSC. Existing research generally focuses on two DSC mechanisms: 1) blockchain-based data validation and 2) fingerprint-based anomaly detection. Blockchain validation and fingerprint anomaly detection controls are shown to be effective in ensuring data integrity and for intrusion and anomalous behavior detection functions.**

This research notes the existence of a void in the assessment and understanding of how combinations of these DSC mechanisms affect ICS networks. To fill this void, this research test combinations of DSC mechanisms in conjunction with standard ICS functions. To do this, an ICS model system is created to emulate the operations of a power substation. The power substation is selected for this research due to its commonality in real-world applications and its simplicity. The model designed in this work includes PLCs, RTUs, a database, and a SCADA controller.

From this design, three Docker networks, a Python scenario script, and a Python security evaluation script are created to emulate, operate, and test the model system. The first network is an un-secured control network comprised of PLCs, an MQTT broker, a database, and a SCADA controller. The second network is a comparison network comprised of the same components but secured by encrypted traffic and a VPN. The third network is the experimental network is comprised of the same components, and implements both a blockchain logging validator and a fingerprint-based anomalous behavior detection server. These two DSC controls are selected due to their prevalence in the existing literature.

System testing reveals that the control system, as predicted, fails all security testing. The comparison network indicated the existence of multiple security flaws, including data integrity and anomaly detection failures, but passed data-in-transit testing. The experimental network passed all data integrity checks and successfully detects anomalous behavior and intrusions, but fails data-in-transfer testing. These results indicate that the combination of traditional security models with combined DSC mechanisms provide a more robust security stance for ICS networks.

I. Introduction

Securing critical infrastructure assets in distributed industrial control systems (ICS) is difficult [1] [2] [3]. This difficulty arises from two competing requirements in critical systems: they *must* efficiently use computationally limited devices *and* they must be secure against attack and failure [3] [4]. On the one hand, a critical system must be fast, accurate, efficient, and robust. On the other hand, if the system is attacked or fails in some way, property may be damaged, people may die, and society may suffer*must be secure* [3] [4]. Most security controls exhibit computationally intense algorithms that, if run on limited computing devices, can cause the system to slow down or crash. In this way, these systems *require* security, but the act of adding security controls to these systems becomes a *very real threat* to the security of the system itself.

These two competing requirements present ICS designers and engineers with a dilemma: they must create a system that is fast, efficient, and functional using equipment with little computational power while simultaneously implementing robust security controls that protect the system without degrading the speed and efficiency of the network [4]. Amid such a dilemma, designers are often reduced to a single choice: function *or* security. When designers face such an ultimatum, always supersedes security. A system that implements robust security measures yet lacks any real function is useless.

Choosing function over security leaves many modern critical systems open to several common security attacks [3] [4] [5]. For example, given a system where remote access must be achieved as quickly as possible, the implementation of time-consuming login protocols can be the difference between the loss of thousands of dollars, the damage to expensive equipment, or even the loss of life. In such an instance, designers will intentionally choose to avoid authentication mechanisms that are commonly used to prevent intrusions [3] [4].

Using current technologies, traditional security controls and methodologies simply do not meet the stringent computational requirements enacted by such limited systems [2]. Many critical systems simply choose to avoid or accept the risk of cybersecurity attacks through avenues that can easily be secured with encryption, authentication, and access controls. This reality necessitates a comprehensive and robust search for alternative security mechanisms; mechanisms that work symbiotically with the distributed nature of distributed ICS networks.

Fortunately, established cryptographic concepts and structures offer promising solutions for these distributed environments. A cryptographic structure is the combination of cryptographic primitives into a new cryptographic mechanism. One such cryptographic structure commonly used today is the blockchain. A blockchain is a distributed record of data that uses data hashes, digital signatures of the various entities operating within the blockchain environment, and a linked list data structure [6]. The distributed nature of the blockchain makes it compatible with most types of distributed networks. The distribution-friendly nature of blockchains makes such a cryptographic structure interesting in the context of distributed ICS networks. Other cryptographic structures and methods, like fingerprinting and distributed authentication, present the same potential benefits if applied to distributed ICS networks.

This work aims to investigate alternative security mechanisms for distributed industrial control systems, specifically those based on distributed cryptographic architectures.

To carry out this work, the following research questions are answered:

1) **RQ.1**: How can Distributed Industrial Control Systems be designed for security using distributed cryptographic security controls?
2) **RQ.2**: What effect, if any, do these combined mechanisms have on the security stance of ICS systems?

By addressing these questions, this research can determine the potential advantages of implementing distribution-friendly security mechanisms in distributed industrial control systems. Given any benefit is discovered, even if marginal, such an improvement can potentially provide real-world benefits to critical infrastructure systems by augmenting the system's security profile.

The remainder of this work will be organized as follows: Section II details the background and foundational information needed to understand industrial control systems and operational technologies. Section II also explores existing works through a short literature survey and classification. Section III will provide an emulation system design methodology, purpose, requirements, architecture, and design. Section IV will outline the software used to emulate the system and explore the configuration of the test system. Section V will analyze the testing regime for the emulated system. Section VI will discuss the outcomes and how the research questions were answered. Finally, Section VII will conclude the paper with a summary of the findings.

II. Background

A. Industrial Control Systems

An Industrial Control System (ICS) is a cyber-physical system that uses programs to control industrial processes, generally through sensing and actuating [7]. These systems use a device called a Programmable Logic Controller (PLC) to carry out cyber-physical interactions in the industrial environment [8].

Most ICS networks operate within a distributed network. These networks connect distributed devices such as PLCs to a central control hub. These centralized control hubs, called Supervisory Control and Data Acquisition (SCADA) systems, use a series of Remote Terminal Units (RTUs) to interact with devices on the distributed network [9]. These SCADA systems encompass both the hardware and software assets implemented to carry out the SCADA system operations.

Most distributed ICS networks using SCADA system architectures operate in hard-real-time environments. Real-time environments are broken into two types: 1) soft-real-time and 2) hard-real-time. In hard-real-time environments, the tasks for which the system is responsible must be completed in a specific period of time [10]. In these systems, the operation *cannot* be late. If, at any point, the response of the system to some physical stimuli extends beyond the specified reaction time, the system fails.

SCADA systems in hard-real-time environments are not arbitrarily limited by the constraints of engineers or developers. Instead, they serve the whims of the physical domains to which they are tethered. For example, in the instance of a power substation, if voltage or current exceeds proper limits, the system must act quickly to break the flow, otherwise, collateral damage can be incurred.

Operational Technologies (OT) comprise any system in which operational constraints outweigh information requirements. Generally, OT is used interchangeably as a means to refer to ICS networks [11].

B. Decentralized Security Controls

A blockchain is a linked list that uses a hash address as a pointer to the next node [12]. A hash is nothing more than a one-way low-cost algorithm that converts any number of input bits to a fixed number of output bits [12]. Blockchains are built as a means to provide immutability in time-scale transaction histories [12]. This is accomplished by hashing the data, signing it, and adding this hash as a reference from the previous block. In this way, the blockchain can record data in a time-linear fashion that cannot be changed after being added.

Digital signatures connect a specific cryptographic key to one particular digital document [12], similar to how a handwritten signature links to a physical document. The process involves creating a hash of the document's contents, encrypting this hash with the private key from a public/private key pair, and attaching the resulting encrypted data to the document. When a user wants to verify the signature, they use the corresponding public key to decrypt the hash. If this decryption succeeds and the resulting hash matches the calculated hash of the document, this proves two things; the document has not been unaltered since signing, and the signature was created using the associated private key. Digital signatures establish non-repudiation, meaning they provide cryptographic proof of a user's interaction with data, preventing them from later denying that interaction occurred. [12].

Code Signing is much like digital signatures. The hash of a codebase is created and signed by the entity producing the code using a private key. When completed, this digital signature is attached to the published software or code package, and the entity makes its public key available alongside the downloadable program. To confirm the software's authenticity, users can perform a simple verification process wherein they decrypt the signature using the public key, generate a hash of the downloaded executable, and then check if this newly calculated hash matches the decrypted signature value [12].

Digital fingerprinting is the process by which the data or physical attributes of a device are used to generate unique identifiers for different hardware and software applications [13]. Fingerprints can be used to identify devices or recognize abnormal behavior. By observing heat, RF emissions, humidity, and other physical attributes of a device, an identifier unique to each device can be generated such that the device and only the device can produce the ID. Fingerprints can also be used to identify common patterns in device functionality [13].

C. Related Work

To better understand the state of ICS networks and distributed security controls, a literature review is conducted. From this short survey, emergent classifications are documented, and each work is classified. A synthesis is then produced, providing an overview of the findings.

D. Literature Review Methodology

This research uses a three-step literature review methodology to generate a literature corpus for further analysis. The three steps used are: 1) literature search, 2) literature selection, and 3) literature synthesis.

The literature review utilizes a set of keywords combined into logical search strings. These strings are applied to two specific databases. The query results are analyzed, and literature in the results that meet the selection criteria is added to the research corpus. Upon completion of the corpus, all the added literature is analyzed, grouped, categorized, and discussed with respect to relevance to the topic of this research.

E. Literature Review

After conducting the systematic review of the literature described above, nineteen (19) relevant pieces of literature were added to the research corpus. From these works, the following three (3) categories emerged: 1) Data Integrity Protection (DIP), 2) Device Fingerprinting (DF), and 3) Decentralized System Design (DSD). In the DIP category, two (2) sub-categories were identified: 1) Centralized Protection and 2) Decentralized Protection. In the DF category, two subcategories were identified: 1) Anomalous Behavior Detection and 2) General Fingerprinting. In the DSD category, two (2) categories emerged: 1) Supporting Decentralized System Weaknesses and 2) Combining Decentralized Protections.

Of the nineteen (19) works analyzed in this literature review, ten (10) fell into the DIP classification. Of the 10 in the DIP classification, three (3) were added to the Centralized Protection classification and seven (7) to the Decentralized Protection classification. Six (6) of the works in this review were added to the DF classification. Of the six (6) works in the DF category, three (3) were added to the Anomalous Behavior Detection sub-category and three (3) were added to the General Fingerprinting sub-category. The remaining three (3) works were added to the DSD category. Of the three (3) in the DSD category, one (1) was added to the Supporting Decentralized System Weaknesses sub-category and two (2) were added to the Combining Decentralized Protections sub-category.

1) Data Integrity Protection: Relating to Centralized Data Protection, Colelli et al. [14] propose a blockchain ledger associated with an ICS historian to provide immutable data tracking for all PLC-related data via a data integrity scanner on the blockchain. Davis et al. [15] propose a blockchain-based traceability solution where all parts and processes of a given manufacturing system are logged, hashed, and appended to the blockchain for quality assurance validation. Schorradt et al. [16] chose to carry out a design akin to Davis et al., but rather than a novel-private blockchain, the researchers chose to add the data to the public Ethereum blockchain. While the application created by Schorradt et al. worked, the use of the public Ethereum blockchain indicated prohibitively slow speeds for real-time operating systems.

In their research on Decentralized Data Protection, Choi et al. [17] recommend implementing a distributed ledger architecture using blockchain technology that simultaneously enhances data security and provides tamper-resistant operational record-keeping within nuclear power plant environments. Otte et al. [18] propose a blockchain for process-level traceability in the mixing of battery chemicals as a means to verify compliance with quality requirements and chemical regulations. Parvizimosaed et al. [19] present a decentralized ledger storage scheme that allows PLC data to be stored at the edge with the PLCs to resist ransomware attacks. Garrocho et al. [20] present a novel blockchain-based access control mechanism for cloud-based Industrial Internet of Things (IIoT) devices housed at the edge for faster and more secure PLC authentications. Jadidi et al. [21] carry out a blockchain similar to other works in this category, but add a deep learning layer to help identify anomalous ICS behaviors. Kirkman et al. [22] provide a ransomware-resistant design that relies on OS-file locks and the continuous running nature of blockchain software to both resist and detect ransomware intrusions. Hayes et al. [23] created a ground-up Raspberry Pi-based communication ledger for IIoT-based decentralized data storage.

2) Fingerprinting: In the area of Anomalous Behavior Detection, Cook et al. [24] introduce a technique for PLC fingerprinting that utilizes each PLC's memory contents to compare memory patterns against established Memory Mapping parameters. Formby et al. [25] present a physics and timing-based fingerprinting mechanism where residual signals and timing artifacts are measured for each PLC to accurately detect anomalous behavior. Lu et al. [26] present a novel noise reduction scheme for the identification of time-based anomalies in IIoT traffic.

Regarding General Fingerprinting approaches, Roy et al. [27] demonstrate a fingerprinting technique that operates externally by monitoring power consumption patterns. Their method records the electrical signatures of PLCs during normal operations and then uses these baseline measurements to identify deviations that may indicate suspicious activity. Keliris et al. [28] present a system where the application of Modbus functionality allows the system to predict different PLC models. Kumar et al. [13] conduct a comprehensive analysis of current advancements in various fingerprinting techniques and methodologies.

3) Decentralized System Design: Concerning the mitigation of recognized vulnerabilities in distributed systems, Garracho et al. [29] propose a failure model to help security professionals patch known weaknesses in IIoT systems enabled with decentralized security.

Relating to the combination of decentralized protections in industrial control systems, Kannelonning et al. [1] categorizes security protections in Norwegian industry, and Hosen et al. [30] design a prototype decentralized security system for an industrial control system.

F. Literature Review Discussion

Based on the literature review carried out in this research, the study of blockchain and decentralized functionality in industrial control systems is not new. Of the nineteen works covered in this review, ten fall into the blockchain or decentralized ledger classification. Although approaches were divided between centralized and decentralized methodologies, a common element across all studies was the innovative implementation of blockchain-inspired technologies within industrial environments, primarily serving distributed data storage needs or secure record-keeping functions.

In addition, device and anomaly identification by fingerprinting represents an established rather than innovative approach. Despite only six works in this study examining fingerprinting techniques, the methodology is thoroughly researched and documented in existing literature. Within industrial control systems, fingerprinting predominantly employs power consumption measurements, physics-based analysis, timing-based evaluation, or memory-based examination to detect unusual behaviors in PLCs or ICS components. Beyond anomaly detection, fingerprinting also serves as an effective mechanism for unique device identification.

This review found that the lack of information lay more in the observations of existing decentralized ICS research. Only three works fell in the Decentralized System Design classification. Of these three, each pushed at the problem from a different angle. Garracho et al. [29] simply tried to understand existing weaknesses, and Kannelonning et al. [1] canvassed the Norwegian industry landscape for security controls. Hosen et al. [30] presented a novel combination of several decentralized ICS protocols to create a holistic design for a possible ICS network.

The findings of the literature review are presented in Figure 1.

III. System Design

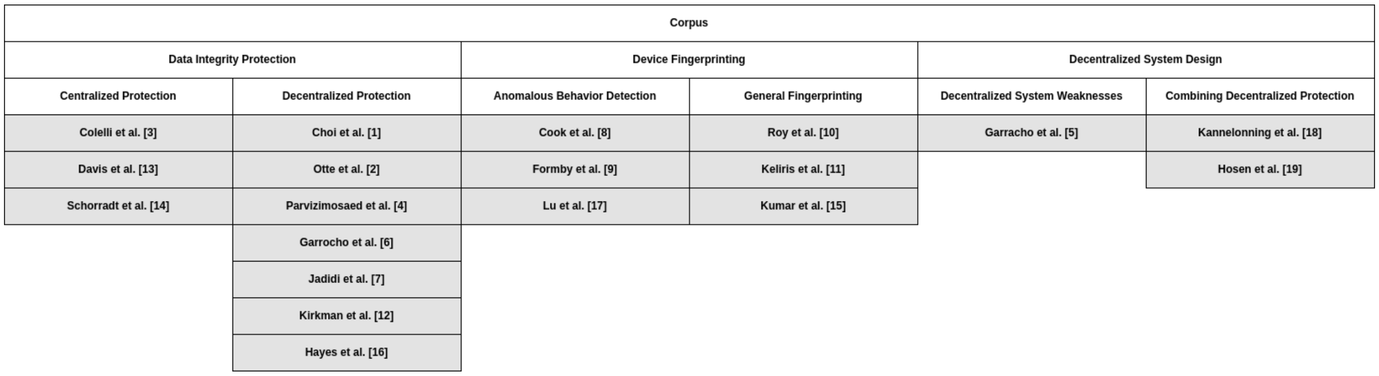
A. Model System Function and Operations

1) Cyber-Physical Systems: All ICS networks serve and support real-world operations [31] in systems called Cyber-Physical systems [31] [3]. The operations of these systems embed ICS devices in functional environments where they are tasked with interacting with real-world sensor input and actuators. Due to ICS networks' mechanical and Cyber-Physical nature, each ICS serves a very well-defined function. This function, served by the system, is the most important service the system must fulfill. These functions cover a wide range of operations such as assembly-like robotics control, oil field operations, power substations, nuclear power applications, hydroelectric dams, etc.

The computing devices in the Cyber-Physical system must control pumps, arms, robots, relays, motors, locks, and more, each serving an innately physical function. In each instance, where a human once operated a physical device or interacted with a physical system, a computer now performs the same tasks [31]. The benefits of automation are three-fold: 1) no longer are humans put in the same dangers as they once were, 2) operations can be streamlined with faster reaction times, and 3) systems can be easily scaled up and down. Alongside such notable benefits, the integration of computer controls in Cyber-Physical systems brings several drawbacks; if a computer fails or operates too slowly, the system can experience catastrophic failures, sometimes even leading to the loss of life.

The connection between the physical function, the industrial design serving the function, and the ICS system makes each ICS network unique [1]. Even across the same application domain, no two OT or ICS networks will be the same. One system may serve an oilfield in Wyoming and another in Texas, and, while they are both oilfields, the topography, landscape, accessibility, weather, and spatial limitations make both systems completely different.

Due to the unique function of every ICS network, in experimentation an explicit definition of the Cyber-Physical function a test system will be responsible for must be defined. Without an adequately designed and defined industrial function, the benchmarks and limits of the system are completely unknown. Without understanding how the system must perform, the integration of experimental security controls cannot be adequately tested for impact. While basic system requirements can indicate general functionality, accurately measuring how well



Corpus					
Data Integrity Protection		Device Fingerprinting		Decentralized System Design	
Centralized Protection	Decentralized Protection	Anomalous Behavior Detection	General Fingerprinting	Decentralized System Weaknesses	Combining Decentralized Protection
Colelli et al. [3]	Choi et al. [1]	Cook et al. [8]	Roy et al. [10]	Garracho et al. [5]	Kannelonning et al. [18]
Davis et al. [13]	Otte et al. [2]	Formby et al. [9]	Keliris et al. [11]		Hosen et al. [19]
Schorradt et al. [14]	Parvizimosaed et al. [4]	Lu et al. [17]	Kumar et al. [15]		
	Garrocho et al. [6]				
	Jadidi et al. [7]				
	Kirkman et al. [12]				
	Hayes et al. [16]				

Fig. 1. Literature Review Results

new DCSs effect the system can only be achieved when tested in specifically designed industrial environments. Knowing that any test system must have an explicitly defined system purpose, this work outlines an industrial system function below.

2) System Model - Industrial Function: The testbed system used in this work will emulate a Power Distribution Substation (PSS). A Power Distribution Substation is nothing more than a physical facility that distributes high-frequency alternating current to different users in the system. Generally, power substations will step down high-frequency power feeds, divert them to low-voltage power lines, regulate the voltages coming and going, and ensure all customers have power. If any issue arises, the safety functions of the substations will automatically operate relays and breakers to protect the system from physical damage.

There are three primary sections of a power substation: 1) Incoming Transmission, 2) Power Routing and Conversion, and 3) Outgoing Distribution [32]. The incoming transmission section is comprised of incoming high-voltage transmission lines. The distribution section is comprised of outgoing low-voltage transmission lines. The power routing and conversion section holds the majority of the PSS devices. Devices in the system include: 1) lightning arresters, 2) air-break switches, 3) step-down transformers, 4) oil-based circuit breakers, 5) voltage regulators, 6) distribution busses, and 7) control housings [32].

The power substation model used in this research is chosen due 1) to the essential nature of the system and 2) the simplicity of the implemented design. As stated above, there are many different types of Cyber-Physical ICS networks, however, the complexity of many of these networks makes the modeling and emulation of the network difficult to achieve. The power substation is chosen due to the ease with which the system can be modeled, and due to the essential nature of substation systems in every town in America.

3) System Model - Industrial Design: The system we will be using is outlined in Figure 2 below:

In Figure 2, the high voltage lines denoted in orange carry high voltage from the incoming feeders through to the outgoing feeders. In the transfer, the voltage moves from a breaker on the high-voltage side through a step-down transformer and a lower voltage breaker before hitting the output bus and moving into the outgoing feeders. Each breaker is controlled via a PLC, and each PLC is connected to the breaker control module. A set of relays is used to determine voltage irregularities and signal the control system to break the circuits.

The notable components in the system in Figure 2 are the 5 PLCs controlling the voltage flow, the breaker control center (RTU), the SCADA control system, and the logging and function databases. The test system will correspond with the PLCs, the MQTT data, the SCADA system, and the databases present in the design above.

B. System Requirements

Industrial Control Systems (ICS) exhibit stringent requirements. ICS networks, comprised of Operational Technology (OT), have several design considerations based on functionality that must be met. Based on NIST SP 800-82r3 [33], the following seven (7) considerations must be taken into account when designing OT-based ICS: 1) safety, 2) control timing requirements, 3) geographic distribution, 4) hierarchy, 5) control complexity, 6) availability, and 7) impact of failures.

Based on the considerations above and the requirements stated in NIST SP 800-82r3 [33], ICS networks must meet timeliness and performance, availability, risk management, physical process, system operation, resource constraint, communications, change management, managed support, component lifetime, and component location requirements.

Functional

Based on the functional requirements stated above, the system proposed must:

1) Synchronize hardware and software assets according to the specific time-critical functions of the system
2) Limit latency, delay, and jitter in communication circuits to accepted tolerances
3) Detect unsafe states and transfer from unsafe to safe with as little human intervention as possible

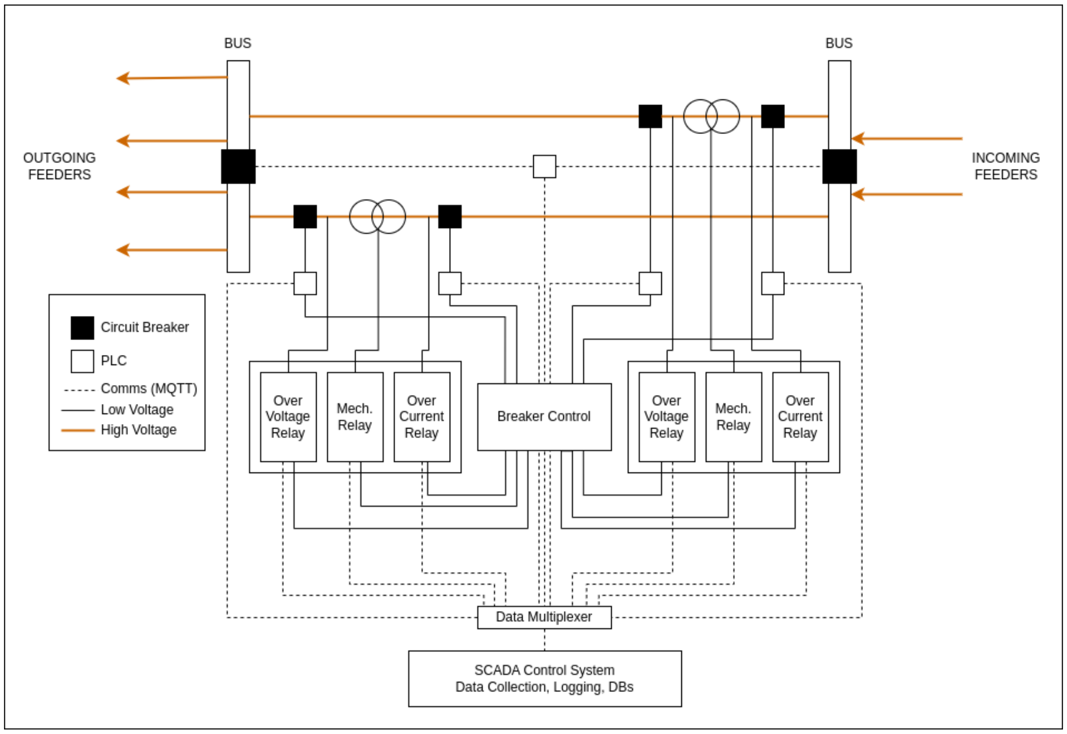


Fig. 2. Clustering: System Design

4) Facilitate bi-directional communication between connected devices
5) Carry out the physical movements needed to both maintain a safe state and carry out system objectives
6) Operate within the constraints of the PLCs and other network devices in the system
7) Exhibit the simplest control complexity possible to meet system operation objectives
8) Coordinate widely distributed hardware devices

Non-Functional

In service of the functions above, the system must also log all access requests, all successful accesses, all data transfers, and all program updates.

C. System Assets, Threats, and Threat/Asset Groupings

1) Selection Methodology and Theoretical Framework: To develop the classification and pairing methodology for threats and assets in Distributed Control Systems(DCS), established frameworks were referenced, and domain-specific considerations for distributed architecture were considered. This section explains the methodological approach used in classifying and pairing assets and threats in DCS environments.

2) Asset Classification Criteria: The asset classification framework was developed using five domain-specific models that span the technological and organizational stack of DCS environments. This classification methodology was derived from an analysis of Industrial Control System(ICS) architecture and was adapted for distributed topologies:

1) **Hardware Assets**: Include field-level devices (PLCs, RTUs, IEDs), human interface equipment (HMI stations, engineering workstations), and other computational infrastructure (historian and control servers, etc.).
2) **Software Assets**: Include operational technology applications (SCADA, DCS software), environments used for development (PLC programming platforms), and infrastructure used as support software (device drivers, alarm management systems, etc.).
3) **Data Assets**: Include information elements considered essential to system operation. This includes dynamic assets (process variables, historical data, etc.) and static assets (configuration files, control logic, etc.).
4) **Communication Assets**: Infrastructure used for connectivity encompasses physical communication media and logical protocols. These were selected to represent the heterogeneous nature of industrial networks that span legacy serial protocols to more modern Ethernet-based standards and some wireless technologies.
5) **Human and Organizational Assets**: Include operational context and governance frameworks for personnel, policies, and procedural elements.

3) Threat Criteria Development: The threat classification methodology used a diversified approach referencing well-known, established frameworks such as MITRE ATT&CK for ICS, IEC 62443 threat modeling, and NIST SP 800-82 [33] security guidelines. Final threat categories were selected based on the following criteria:

1) **Threat Actor Perspective**: Categories decided upon

addressed intentional attacks, such as cyber attacks and insider threats, as well as unintentional events such as operational, physical, and environmental threats.

2) **System Lifecycle Orientation**: The threats in this section map to system lifecycle phases. Including architecture and design issues and threats, operational threats, and governance and compliance issues.
3) **Attack Vector Categorization**: This section includes threats organized by the attack vector potentially used by the threat actor, including data and communication threats, supply chain threats, etc.

The final threat criteria presented represent a comprehensive model for ICS security that extends beyond simply technical vulnerabilities to include organizational, procedural, and governance facets, which are considered critical for a holistic security analysis of distributed control systems.

4) Asset-Threat Pairing: The assets and threats previously listed were paired using a phased analytical process that included historical incident analysis, where a thorough examination of documented ICS security incidents was undertaken to identify asset-threat relationships. This was followed by examining the common attack surfaces found within ICS networks. This examination led to a systematic evaluation of potential vulnerabilities within each asset based on noted interfaces, dependencies, and operational context. Risk-based prioritization was then employed to apply qualitative risk assessment methodologies to focus on high-impact and high-likelihood scenarios. These scenarios were particularly focused on critical operational technology components. The final analytical steps taken involved ensuring comprehensive coverage. The threat-asset pairings were cross-referenced against known and established security frameworks to ensure there were no omitted vulnerability classes. The pairings presented represent and demonstrate the complex interdependencies seen within DCS environments where threats have the potential to surpass traditional security domains.

5) Control Function Classification: The control function framework for distributed systems aligns with the hierarchical framework defined by the ISA-95/IEC 62264 standards for industrial automation, with specific modifications to accommodate the characteristics of distributed architectures. This framework encompasses several principal functional domains. Process control functions constitute the foundational and advanced control loops that interface directly with physical processes. Safety and alarm controls serve as protective mechanisms designed to maintain safe operating conditions. Supervisory and optimization controls represent higher-level functions to enhance overall system efficiency and performance metrics.

The framework further incorporates communication and network controls, essential for enabling data exchange and connectivity across distributed components, and redundancy and fail-over controls, which ensure operational continuity and system resilience in the event of component failures. Additionally, power and energy controls are included to manage the electrical infrastructure and resource allocation, while quality and production controls oversee consistency in output and enforce quality assurance protocols.

This structured framework facilitates a comprehensive and systematic analysis of security considerations across the various control functions within distributed operational environments.

6) Industrial Control System(ICS) Requirements and Security Considerations: Given their role in supporting critical infrastructure, Industrial Control Systems must compose stringent requirements that directly influence the classification strategies for both assets and threats. NIST SP 800-82 Rev. 3 outlines several design considerations deemed foundational for incorporation into ICS architecture. These principles significantly informed the classification methodology for this project.

One of the primary considerations is the timely detection of unsafe conditions and the ability to transition any unsafe system into a secure operational state. This detection capability directly corresponds to the implementation of protective controls and alert mechanisms contained within a broader classification of distributed control functions. Identifying and categorizing communication-related assets and the specific threats targeting synchronization services directly informed the need for precise, time-sensitive synchronization across ICS networks.

The broad physical distribution of ICS networks across multiple geographic locations further shaped the approach to classification, especially regarding threats that could reasonably be considered to affect distributed architectures and communication systems. Recognizing that ICS deployments can range from highly localized environments to extensive, geographically dispersed networks was key in structuring these categories.

ICS environments are typically centrally managed; this centralized management model influenced the classification of monitoring and optimization functions and had a hand in characterizing threat vectors aimed at hierarchical control systems. Operational process complexity also guided the categorization of functions, with an emphasis on appropriate oversight and coordination in distributed contexts. System resilience features such as backup and recovery components play a critical role and further informed the classification of alternative pathways and their respective vulnerabilities.

Operational threats and failure modes across all assets informed the assessment of potential consequences and played a central role in defining these threats and failure modes. The asset classification framework resulting from this assessment was aligned via functional requirements such as timeliness, operational continuity, risk mitigation, and secure data exchange, which are fundamental to successful ICS deployment and performance.

Finally, the security principles outlined in NIST guidance provided essential validation for the threat classification model. Specifically, the imperative that security controls must not compromise the real-time responsiveness, continuous availability, or safe operation of ICS environments served as a core criterion in shaping the threat categorization, ensuring alignment with the unique demands of operational technology systems.

D. Traditional and Decentralized Security Controls

Modern ICS networks implement several common security controls. Due to the functional and environmental constraints present in ICS networks, the most common security controls are encryption, authentication, and identity management [3] [2] [1]. Though more comprehensive security mechanisms have been tested in Cyber-Physical systems and ICS networks, the computing overhead they require makes them more of a risk to the system than a benefit [2] [10].

While common security controls do provide a functional solution to ICS security, the unavoidable computing and environmental restrictions in ICS networks keep the security stance of most ICS networks "only-just" secure enough [2] [8]. Most modern critical systems use ICS networks and Cyber-Physical systems as an implementation mechanism [2]. The critical nature of these systems requires a deep exploration of plausible security improvements. Though common security controls work, there are unexplored avenues for potential security improvements.

This research aims to explore the use of decentralized security controls in ICS networks as a contrast to commonly used security mechanisms. Decentralized security controls are any security control that allows the decentralization of security mechanisms. In a true decentralized security environment, each device would manage a piece of the security picture, effectively distributing the work required to keep the system secure. The most common examples of decentralized security primitives are blockchains and device fingerprinting.

Blockchains are used to create immutable logging mechanisms by which a system can verify that the data it uses has not been tampered with [12]. Blockchains have been tested numerous times in ICS networks [34] [14] [15] [16] [17] [18] [19] [30]. While blockchains are not new to ICS network security, most research using blockchain primitives examines the blockchain alone in the context of a Cyber-Physical system, not in conjunction with other security mechanisms.

Device fingerprinting is used to analyze system behavior for anomalous device activities [25]. The use of fingerprinting is also not new in ICS networks [25] [13] [24] [27] [28]. Again, most testing concerning the implementation of device fingerprinting in ICS networks attempts to understand how fingerprinting functions in isolation when added to an ICS network. They are not tested in conjunction with other decentralized security controls.

Due to the existence of available research concerning both blockchain- and device fingerprinting-based security mechanisms in ICS networks, this research proposes the use of a blockchain-enabled integrity validation server in conjunction with a data-based device fingerprinting server. By deploying these solutions into an ICS network, observations can be made as to the effects, both relating to security and functionality, that these systems have on ICS networks.

E. System Design Methodology

Industrial Control Systems are intrinsically tied to specific environmental restrictions and often costly infrastructure and hardware components [31]. These limitations make research and development on ICS networks difficult, as many researchers do not have the resources to create entire Cyber-Physical test systems. The ICS networks that already exist in the wild are often attached to existing critical applications [31] [1] [2]. Due to their critical importance in social infrastructure, these networks are generally closed to research and development purposes. To change the system during the research process poses far too much risk to the system and thus is generally not allowed.

These limitations make research concerning Industrial Systems difficult. Researchers, instead, need to opt for more flexible and accessible experimental domains. Containerization is a type of virtualization that allows applications to be packaged such that they run interdependently of an operating system [35]. These containers enable many different functions and processes when creating networks of interconnected applications. These functions can be combined to create systems of containers that can be used to model and emulate functional systems. [35] [36]. The use of these containers provides flexibility, ease-of-access, affordability, and less risk when testing [37] [36].

By using containerization to emulate the devices used in the Cyber-Physical systems involved with industrial control networks, researchers can create robust, consistent, and accurate models for testing and analysis of ICS modifications and upgrades. Further, container applications can be created with custom code that can be programmed to emulate nearly any device that can be added to an ICS network. Such cheap, flexible, and accessible characteristics make containerization a strong candidate for ICS research and development [35].

The most popular containerization service currently is *Docker*. Docker uses a centralized hub to serve application code packages called images to Docker systems worldwide. To create a custom image, a developer can write out emulation or modeling code, package the code into an image, and push the image to the centralized repository. To use the image, it only needs to be pulled from the repository and booted. When booted, Docker images are called containers. These containers can be comprised of one or many different Docker images and can be configured to communicate internally, much like a traditional IT network. Developers simply create a Docker Compose file that specifies the services and connection specifics and use Docker to run the Compose file. The result is a system that can be modeled to carry out nearly limitless possible functions.

This research uses Docker, via several existing ICS device images and several custom images, to model and emulate the ICS system proposed in this work. By using Docker, the flexibility required in research can be achieved while providing accessible, consistent, and accurate system modeling functionalities.

IV. System Emulation

Based on the design presented in the previous section, an emulation environment is constructed to conduct testing.

A. Architecture

To carry out the experiments and testing required in this research, three networks are designed based on the system design architecture outlined in the previous section. These three networks are: 1) an unsecured ICS network, 2) an ICS network secured by common security controls, and 3) an ICS network secured by two decentralized security mechanisms. These three networks are chosen to provide useful comparisons between a control system (the unsecured ICS network), a comparison network (the ICS network using common security mechanisms), and the proposed experimental system.

Based on the proposed design architecture from the previous section, three *Docker-based Network* designs are created. Docker is used for emulation due to its flexibility, ease-of-use, and access, and its ability to model real systems. More detailed rationale concerning the choice of Docker in this work can be found in Section III, part E.

1) Docker Network Designs: The control network shown in Figure 3 is deployed without any security control mechanisms. The network contains three PLCs, an Ignition SCADA control system, a Mosquitto MQTT data broker, an Influx Database, and a Grafana Data Analysis Dashboard. These devices are chosen to represent the PLC-to-SCADA functionality present in Figure 2.

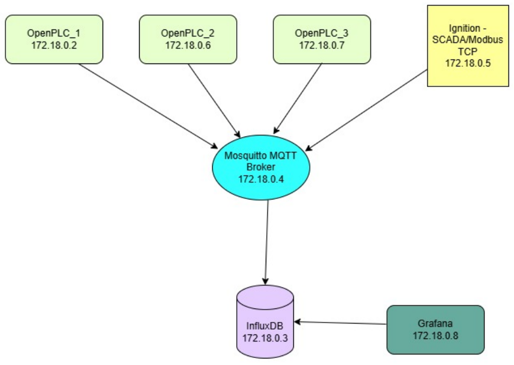


Fig. 3. Clustering: Control Network Design

The comparison network represented in Figure 4 uses encryption and a VPN to protect the data in the system. The network is comprised of three PLCs, a Modbus TLS Gateway, an OPC-UA Server, an Ignition SCADA Server, a Mosquitto MQTT data broker, an Influx Database, and a Grafana data analysis dashboard. These devices represent the PLC-to-SCADA functionality present in Figure 2.

The experimental network represented in Figure 5 uses four PLCs, a Mosquitto MQTT broker, an Ignition SCADA server, a custom fingerprinting server, an Influx database, and a blockchain validation server. These devices represent the PLC-to-SCADA functionality present in Figure 2.

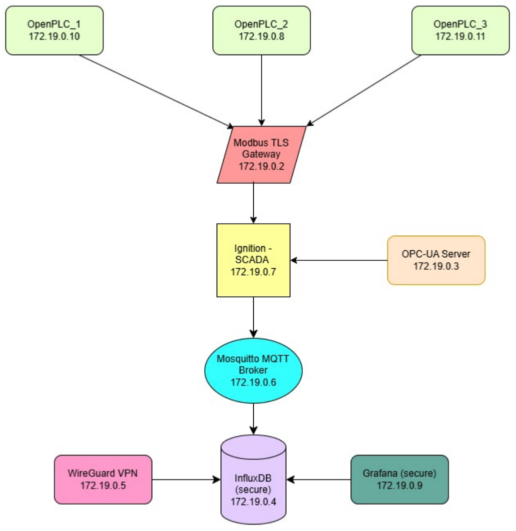


Fig. 4. Clustering: Comparison Network Design

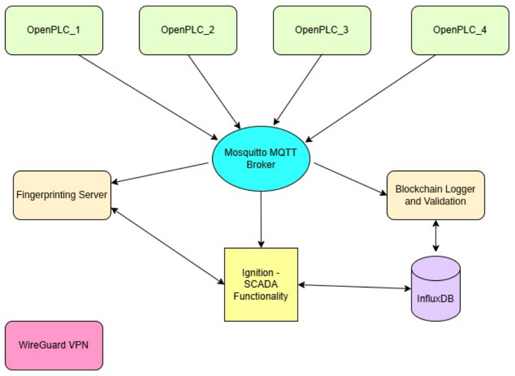


Fig. 5. Clustering: Experimental Network Design

B. Docker Networks

This section summarizes the successful development and deployment of a virtual Docker-based ICS.

1) Technology Stack: The following Docker images and technologies are integrated into the model system to achieve the stated goals:

- *PLC (OpenPLC)*: Programmable Logic Controller emulator (Modbus TCP).
- *MQTT Broker (Eclipse Mosquitto)*: Lightweight message queuing for inter-container communication.
- *InfluxDB*: Centralized time-series database for metrics storage.
- *Grafana*: Real-time metrics visualization dashboards.
- *SCADA (Ignition)*: Supervisory Control and Data Acquisition for system monitoring.
- *Modbus-TLS Gateway (Stunnel)*: Securing Modbus communications.
- *VPN Gateway (WireGuard)*: Encrypted secure tunneling between container networks.

- *OPC-UA Server (Open62541)*: Open standard protocol for interoperability in automation.
- *Blockchain Integrity Server*: Verifying data integrity and authenticity using blockchain technology.
- *Fingerprinting Server*: Authentication service for validating network clients.

2) Implementation Details: Each Docker environment was configured using YAML files, facilitating clear service and image definitions, network configurations, data volumes, startup dependencies, and secure communication setups. Below is a breakdown of each Docker environment.

1) **Non-secure environment (control):**
   - *Purpose*: A reference environment for benchmarking without additional security.
   - *Containers/Technologies*: OpenPLC, MQTT Broker, InfluxDB (no encryption), Grafana dashboard (no encryption), SCADA system (Ignition)
2) **Secure environment (comparison):**
   - *Purpose*: Implement standardized secure communications, encryption, and protocols.
   - *Containers/Technologies*: OpenPLC with Modbus-TLS, MQTT Broker (TLS-enabled), Secure InfluxDB (TLS encryption), Grafana with secure connections, SCADA system (Ignition), Modbus-TLS Gateway (Stunnel), VPN Gateway (WireGuard), OPC-UA Server (secure communication)
3) **Novel DSC environment (experimental):**
   - *Purpose*: Implement novel-based secure communications, encryption, and protocols.
   - *Containers/Technologies*: OpenPLC (standard Modbus TCP), MQTT Broker, InfluxDB for centralized metrics, Grafana dashboards, SCADA system (Ignition), Blockchain Packet Integrity Server (ensuring data integrity), Fingerprinting Server (anomaly detection)

3) Deployment Automation: A Bash deployment script (deploy.sh) is used to automate container life cycle management, environment setup, and token handling for secure access to InfluxDB. Key features of this shell script include:

- *Automated cleanup:* Removing existing Docker entities before deployments.
- *Docker image updates:* Ensuring the most recent images are always used.
- *Docker environment implementation:* Runs docker-compose commands against each YAML file.
- *InfluxDB token management:* Retrieval, validation, and persistence of authentication tokens for each environment.

4) Docker Environment Benefits Achieved: By leveraging Docker, the research team successfully achieves all project objectives, creating a highly automated, scalable, secure, and maintainable distributed control system. The seamless integration of YAML configurations, Docker containerization, and automation scripts ensures the system's long-term viability and adaptability. This robust architecture provides a reliable foundation for future research initiatives, industrial security assessments, or deployment in testbed environments that simulate critical infrastructure.

C. Decentralized Security Control Components

To test the use of decentralized security controls in ICS networks, an experimental Docker network is configured, including an example of the two most prominent DSC mechanisms: a blockchain integrity monitor and a fingerprinting server. The blockchain integrity server either adds data to a blockchain or checks the integrity of previously added data [14] [15] [16] [17] [19] [21]. The fingerprinting server evaluates incoming data for a specific operational pattern [13] [24] [25] [28]. If the pattern is observed, it responds with a ”true”. If the pattern diverges, the server will respond with a ”false”.

1) Blockchain Integrity Server: The blockchain integrity server (BIS) is based on the official Python image found on DockerHub. Within this Python image, a server.py script is created that manages connections to and from the server via sockets connections. Sockets are used to create communication between processes and across networks. The use of sockets allows servers to send and receive data. The server file, in its most basic form, simply sets up the send and receive capability and manages the handling of incoming JSON traffic.

The blockchain implemented on the server uses a hashed linked list to provide integrity validation for data in the blockchain. A $Node$ class and a $Blockchain$ class are used to define the blockchain functions.

2) Node: The *Node* class wraps five data attributes: the time as a datetime, the data as a string, a calculated hash of the node as a string, the next Node in the list, and a boolean representing if the node is the head.

The primary function in the Node class is the $set\_hash()$ function. To ensure that integrity is validated from node to node, the hash of the current node *must* implement the hash of the previous node.

3) Blockchain: The *Blockchain* class wraps a simple data attribute called *head* of type *Node*.

The primary functions in the *Blockchain* class are: $add\_node()$ and $verify()$. The $add\_node()$ function extends the blockchain by calling the $set\_next()$ function in the *Node* instance. The $verify()$ function starts at the head of the blockchain, validates each Node for tampering, and, if the queried data is present, returns a $true$ response.

The $add\_node()$ function saves the head into a temporary variable, then iterates through each next Node until it reaches the end of the blockchain. When it reaches the end of the blockchain, a new empty Node is added if the $time$ and $data$ parameters are None. If they are not None, a new Node instance is created with the incoming time and data and the hash of the current Node.

The $verify()$ function first saves the head and the hash of the head Node. It then iterates while the next Node of the next Node is not none and while the data has not been found. If the next.hash ever matches the hash of the incoming data, the data has been validated.

4) Blockchain Integrity Operation: In function, the BIS expects to receive a JSON with three keys: *verify* as a boolean, *time* as a string, and *data* as a string. If the *verify* key is set to *false*, the blockchain will attempt to validate the information in the *data* key.

For each JSON received, the server will respond. The response JSON uses three keys: *verify* as a boolean, *data* as a string, and *result* as a boolean. If the *verify* key is *true* and the data is validated, the *response* will be *true*; else if the data is not validated, the *response* will be *false*. If the *verify* key is *false*, the *response* will indicate if the data was successfully added to the blockchain.

D. Fingerprinting Server

The Fingerprinting Server (FS) implements the same blockchain structure and sockets server functionality. The same functions listed in the Blockchain are used to save, validate, and recover data. The only difference lies in the $verify\_pattern()$ function in the FS server code. In function, the FS diverges from the BIS in its use of the blockchain. Rather than use the chain for integrity validation, it uses the chain to ensure that the appropriate pattern of data is present in the blockchain. If an anomaly is detected, the server responds with a negative result. To test the functionality of the FS, the $verify\_pattern()$ function is used.

The $verify\_pattern()$ starts at the top of the blockchain, iterates until the first state in the blockchain that matches the existing state is found. From this point forward, the transition from state to state is measured and matched until the end of the block is reached. If all blocks match, a $true$ response is returned, else a $false$ response is generated.

E. Network Operation and Functional Scenarios

Using the three networks online (the "baseline" control network, the "encrypted" comparison network, and the experimental network including DSCs), three testing instances, built in Python, generate network traffic that verifies the nominal operations of each environment.

1) Control Network Traffic Generation: The `populate_data.py` script simulates random sensor data from multiple Programmable Logic Controllers (PLCs) in the non-encrypted network. This synthetic data is published to the MQTT broker and written to the InfluxDB time-series database.

The script runs continuously and mimics real-time industrial traffic generation, which is useful for testing ICS infrastructure, data pipelines, and monitoring solutions. The code uses the following Python libraries: 1) paho.mqtt.client, 2) influxdb-client, 3) random, 4) time, 5) json, and 6) socket.

In the script, the $connect\_mqtt(client)$ function attempts to connect the MQTT client to the broker. If the connection fails (e.g., the broker is not up), it will retry every 5 seconds until successful. The $generate\_plc\_data(plc\_id)$ function generates a dictionary representing PLC sensor data. The $publish\_data(mqtt\_client)$ function continuously cycles through three PLCs (IDs 1–3), sending data JSON data and writing values to the InfluxDB. When the script is executed, it first initializes the MQTT client and connects the broker. It then starts the MQTT event loop and beings publishing data using the $publish\_data$ function.

2) Encrypted Network Traffic Generation: The `encrypted_populate_data.py` script simulates industrial data traffic from multiple PLCs and sends that data to both the MQTT broker using TLS and to the InfluxDB via HTTPS. This script is intended for use in the encrypted emulated network, where traffic must traverse encrypted channels to validate TLS handling and authenticated writing. The script uses the same Python libraries as listed in the control network.

In the script, the $connect\_mqtt(client)$ function tries to connect the MQTT client to the broker over port 8883. It includes retry logic with a 5-second wait after each failure, which allows the script to tolerate broker unavailability on startup. The $generate\_plc\_data(plc\_id)$ function simulates the same range of sensor data as the non-encrypted environment. The $publish\_data(mqtt\_client)$ function continuously loops through three PLC IDs, sending JSON data and writing to the InfluxDB with secure token-based writing. When `encrypted_populate_data.py` runs, it initializes the MQTT client, creates an MQTT connection, starts the MQTT loop, and begins publishing data.

The encrypted environment has different security considerations. Namely, MQTT over port 8883 is assumed to use TLS, but no certificates or TLS parameters are specified in this script. The data written to InfluxDB is sent via HTTPS and authenticated with a secure token. Finally, the script assumes that the MQTT broker and the InfluxDB server accept the default client configuration and do not require client certificates or additional authentication headers.

3) DCS Network Traffic Generation: Using direct socket communication and serialized data structures, the ICS Test Client utilizes the `test_runner.py` script to test two novel components in the **blockchain server**, which stores and verifies data entries; and in the **fingerprinting server**, which detects operational anomalies based on pressure patterns. The script uses Python's `socket` and `pickle` libraries to exchange data over TCP, and logs both the execution output and a structured result summary to a timestamped log file.

In the script, the $send\_json\_to\_server(host, port, json\_data)$ function creates a socket connection to a specified host and port, sends a pickled Python dictionary, and returns the unpickled response. The function includes a 5-second connection timeout. The $random\_string(length = 50)$ function generates a random alphanumeric string of the given length (default: 50 characters), which is used to simulate payload data. The $test\_blockchain\_add()$ function sends five `verify=False` messages to the blockchain server, each with unique random data and timestamps. It then collects and returns both the request and response for verification. The $test\_blockchain\_verify(added\_entries)$ function replays the data from the `test_blockchain_add` function using `verify=True` to check if the entries are still valid on the blockchain. The $build\_pressure\_json(pressure\_value, verify = False)$ function creates a dictionary formatted for the fingerprinting server. The `data` key contains a pressure value as a string. The $test\_fingerprinting\_valid\_pattern()$ function sends a known good pressure oscillation pattern to the fingerprinting server, alternating between `verify=True` and `verify=False`. It returns both payloads and responses. The $test\_fingerprinting\_invalid\_pattern()$ function sends a malformed pressure pattern with unexpected values. Each message includes `verify=True`. This validates the fingerprinting server's anomaly detection.

When `test_runner.py` runs, it executes and collects results from the blockchain and fingerprinting tests. The complete results dictionary is sent as an output to both the terminal and a timestamped log file.

F. Security and Attack Functions

1) Security and Vulnerability Testing: The security testing scripts in this project evaluate and compare the security posture of the three Dockerized ICS environments: the baseline environment without cryptographic protections (NoCryptoTest.py), the encrypted environment using TLS (CryptoTest.py), and the blockchain-enabled environment (BlockchainTest.py).

When packaged as Docker containers, these scripts run against these pre-configured containers. Since each environment is configured to operate as an isolated subnet, this ensures consistent testing despite architectural differences.

The testing scripts follow a structured approach, starting with basic connectivity checks through ping tests and port scans, followed by more component-specific tests. The scripts assess PLCs for unauthenticated access via Modbus and EtherNet/IP, evaluate MQTT brokers for unauthorized connections and excessive topic subscriptions, and test databases for proper authentication.

To optimize efficiency, these scripts use a multi-threaded model to allow parallel testing of components, reducing overall testing time while maintaining accuracy. Error handling within the scripts ensures that isolated failures do not affect the entire process, and any missing dependencies are noted in the reports.

Each script tailors the tests to the specific security model of the environment:

1) NoCryptoTest.py focuses on basic security controls, such as authentication and access control, to assess the vulnerability of the environment without cryptographic protection.
2) CryptoTest.py extensively tests TLS implementations, including certificate validity, cipher strength, and encryption enforcement.
3) BlockchainTest.py evaluates the integrity of the distributed ledger, testing unauthorized data submissions and device identity verification mechanisms.

All test results are logged with timestamps and severity classifications (PASS, FAIL, WARN, INFO, SKIP). Upon completion, each script generates a detailed JSON report summarizing test results, highlighting critical findings, and offering insight into security improvements gained through encryption and blockchain technologies.

These scripts translate the theoretical security framework into practical assessments, demonstrating the impact of security mechanisms on system resilience and providing quantifiable evidence of improvements from encryption and blockchain integration.

V. Testing

A. Data Gathering and Results

To gather network traffic from test environments, a packet capture is performed in each environment using the command line packet analyzer `tcpdump`. The command utilizes the following form: `sudo tcpdump -i <network ID> -w /path/to/capture.pcap`.

The nominal traffic generation test for the unencrypted network serves as a means to verify proper connectivity between the PLCs and the MQTT broker, as well as ensure the data is published to InfluxDB. While testing, an observed publish message from Grafana indicates a successful data transmission from PLC 3. This shows a successful network function, but the data in transit over the emulated ICS network is plaintext without any verifications of message integrity. Therefore, since it has no security controls, it fails integrity and functional verifications.

A similar test is deployed on the encrypted network using standard TLS encryption. Of course, since the network data is encrypted in transit, the packet capture will not display the data in plaintext. Verifications using InfluxDB to validate the data received are necessary, but a packet capture will still show the TLS network traffic. A successful TLS handshake and data exchange between PLC 1 and InfluxDB occurs, confirming that the encrypted network has data protections in place, with no way of verifying the integrity of the data sent and received. This network also fails integrity and functional verifications.

Testing the final network and its novel blockchain and fingerprinting functions requires a different approach. To validate the proper operation of the unencrypted and encrypted network, the only requirements were to show that data is sent and received in non-encrypted (plaintext) and encrypted form, respectively. Consideration for the testing of this network is given to the blockchain and fingerprinting functions, and the formatting of the messages these functions were expecting: pickled `JSON` data. Testing not only includes data transmission but also whether the blockchain and fingerprinting functions verify the integrity of the data. A packet capture and log message between the communicating PLCs and the blockchain and fingerprinting instances shows the successful verification and integrity check of the data being sent and received. One glaring issue exists for future improvement: the data is not encrypted.

1) Security and Vulnerability Testing Results: The script outputs illustrate a clear progression in security across the three distinct industrial network environments.

The evaluation of the basic non-encrypted environment revealed notable security weaknesses. Default credentials, such as "admin," were used to successfully log in to PLC1. Several components, including PLC1 and Grafana, had open and unsecured ports. Others were inaccessible, possibly due to configuration errors or firewall restrictions. Security was minimal and relied only on basic password protection with no encryption applied to communications.

When evaluated, the encrypted environment showed enhancements in security architecture through implementing TLS configurations and tighter port access. Even with these enhancements, several components did not pass TLS checks. Security testing was performed via the Dockerized scripts, which included setting up encrypted ports for various components, such as configuring the MQTT brokers to use port 8883 instead of 1883. Network segmentation was confirmed with all anticipated devices in place. However, Grafana was found to lack security headers.

Key issues in the encrypted environment

1) 14 critical failures
2) Missing or improperly configured TLS support on several components

Implementation of blockchain technology helped ensure data integrity and fingerprinting and enabled device identity verification in the third environment. Due to the security enhancements, this environment showed a higher pass rate in security tests and more consistent enforcement of port security.

Key improvements in the blockchain environment

1) Enhanced data integrity verification through blockchain
2) Better detection of device impersonation through fingerprinting
3) Improved authentication mechanisms

The transition from the non-encrypted environment to the blockchain-enabled system shows a security evolution from basic password protection to encryption and finally to data integrity and identity verification. Security implementation throughout the different environments shifts from individual device protection to network security and, ultimately, to a system-wide trust model.

While these results show substantial security improvements from transport encryption to blockchain-based data integrity, implementation challenges persist across all environments. Many components still failed security tests even in the most advanced setup. These failures further highlight the complexity of securing industrial control systems and the need for continuous testing and improvement.

VI. Discussion

To answer RQ1, a preliminary literature review and ICS documentation search are carried out, and the results are analyzed for emergent classifications. From these emergent classifications, a set of decentralized controls emerges alongside a subsection of work focused specifically on DSC from a system perspective. From this review, the use of DSC methods in ICS networks is not new, but lacks significant research efforts.

From this literature search, the most notable and well-researched DSC mechanisms implemented in ICS networks are anomalous behavior detection via data and physical device fingerprinting and blockchain integrity validation. Fingerprinting in ICS networks is used to analyze the physical, digital, and operational patterns in ICS networks for abnormal or erroneous behavior. Due to the predictability in all ICS network operations, such anomalous behaviors in ICS networks are often the result of either an environmental breakdown or a cyber attack. Blockchains are usually integrated into ICS networks to provide immutable logging or data validation. Logs and/or essential data attributes are written into the blockchain, so any log/data retrieval can call the blockchain integrity validation server to verify that the log being pulled has not been tampered with. A blockchain integrity server acting within an ICS logs, validates, and propagates the decentralized blockchain such that local data in the ICS network can be protected against tampering.

From the review completed in this work, distributed industrial control systems can be designed using any combination of DSC controls. These DSC controls can be used in conjunction with standard security protocols or in isolation. From a system-design level, a holistic system architecture can be created that can capitalize on the benefits of traditional security while boosting security from a decentralized point of view.

To answer RQ2, a model cyber-physical system design is created. Within the modeled design, a system model of industrial function is outlined. In this experiment, the emulated design follows power substation design specifications. Within the ICS network, several PLCs control voltages moving across the substation. These PLCs are responsible for tripping breakers, given that the voltage in the system exceeds expectations. After the industrial function is settled, a set of system requirements and functional and non-functional requirements are developed to add granularity and resolution to the model system design. With a system design settled, threats to the system are identified and paired with the assets within the system. Many decentralized security controls are identified from these asset/threat pairs, thus providing mitigating controls for each asset and threat.

From this model system design, a series of emulated ICS networks is created using the Docker emulation software. Three Docker networks are created as follows: 1) a control network (Figure 3), 2) a comparison network (Figure 4), and 3) an experimental network (Figure 5). The control network implements the industrial control system design, but provides no security controls. The comparison network implements the model system design alongside standard security controls for ICS networks as defined by NIST SP 800-82 [33]. Finally, the experimental network implements the model system design with two DSC controls: Fingerprint Anomaly Detection and Blockchain Integrity Validation. Following the methodologies identified in the literature review presented in Section II, this work includes the design and implementation of both a fingerprinting server and a blockchain server as core components of the security infrastructure.

Upon the creation of the emulated Docker systems, a single scenario script is used to emulate the operational functionalities of the designed industrial system. This operational script runs the PLCs, SCADA controllers, Databases, Servers, and Security Mechanisms according to a preset operational pattern. Another corresponding set of security evaluation scripts is created to perform basic security analysis scans that provide clarity as to the security stance of each network. To measure the effect of each security control, system traffic is analyzed via packet capture, and the scenario script collects data as returned from each device in the system.

From these results, each network provided the following insights:

- **Control**: All security tests fail due to the absence of any security controls
- **Compare**: Standard security testing reveals 14 critical failures and improperly configured TLS components
- **Experiment**: DSC security testing provides no data protection in transit, but successfully detects malformed packets, data tampering, abnormal behavior, and boosted authentication functions.

The results of the experiment indicate that both the comparison network and the DSC networks are more secure than the unsecured control network. The comparison network provides much better data protection in transit, but does not provide any protection against anomalous behavior and data tampering. By combining both the comparison network and the experimental network, the security benefits of both networks are combined to create a better security posture for the ICS network.

VII. Conclusion

The results of this work indicate that the use of decentralized security controls in ICS networks is not new, but is underdeveloped. The research in this area generally favors single security control test environments, not system-level security control designs. From existing literature, blockchains and fingerprinting functions are the most common decentralized security controls found in ICS networks. Fingerprinting mechanisms in ICS networks enable anomaly detection by establishing baseline patterns of normal operational functions and comparing current system activities against these expected behavioral profiles. The observed fluctuating patterns suggest the system is operating outside normal parameters, which may indicate either a security compromise or environmental system failure.

A model ICS system design is created to emulate a power substation. Assets, systems, functional and non-functional requirements, threats, and threat/asset pairs are determined for the system. From these specifications, a model system is designed to meet the needs of a power substation. From this model design, three Docker networks are created: 1) a control network, 2) a comparison network, and 3) an experimental network. The three systems are operated via a consistent operational scheme, and observations are collected regarding system security and functionality.

The conclusions of this research indicate that the addition of DSC mechanisms to ICS networks provides a more robust security stance than the use of traditional security mechanisms alone. By adding fingerprinting and blockchain integrity functions, the experimental network, in combination with comparison network functions, provides a more robust security control design.

While DSC research is not new, this research adds fidelity to the understanding of more holistic design activities and indicates that the use of DSC mechanisms in conjunction with standard security controls can provide elevated security posturing for critical systems security.

References

[1] K. Kannelønning and S. Katsikas, "Deployment of cybersecurity controls in the norwegian industry 4.0," in *Proceedings of the 19th International Conference on Availability, Reliability and Security*, ser. ARES '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3664476.3670896

[2] J. C. Knight, "Safety critical systems: challenges and directions," in *Proceedings of the 24th International Conference on Software Engineering*, ser. ICSE '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 547–550. [Online]. Available: https://doi.org/10.1145/581339.581406

[3] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee, and S. K. S. Gupta, "Ensuring safety, security, and sustainability of mission-critical cyber–physical systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 283–299, 2012.

[4] L. E. G. Martins and T. Gorschek, "Requirements engineering for safety-critical systems: Overview and challenges," *IEEE Software*, vol. 34, no. 4, pp. 49–57, 2017.

[5] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez, "A survey of iot-enabled cyberattacks: Assessing attack paths to critical infrastructures and services," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3453–3495, 2018.

[6] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Access*, vol. 7, pp. 117 134–117 151, 2019.

[7] C. M. Poskitt, Y. Chen, J. Sun, and Y. Jiang, "Finding causally different tests for an industrial control system," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 2578–2590.

[8] W. Alsabbagh and P. Langendörfer, "Security of programmable logic controllers and related systems: Today and tomorrow," *IEEE Open Journal of the Industrial Electronics Society*, vol. 4, pp. 659–693, 2023.

[9] M. M. Ahmed and W. L. Soo, "Customized scada system for low voltage distribution automation system," in *2009 Transmission & Distribution Conference & Exposition: Asia and Pacific*, 2009, pp. 1–4.

[10] D. Leinbaugh, "Guaranteed response times in a hard-real-time environment," *IEEE Transactions on Software Engineering*, vol. SE-6, no. 1, pp. 85–91, 1980.

[11] M. Shilenge and A. Telukdarie, "Optimization of operational and information technology integration towards industry 4.0," in *2022 IEEE 31st International Symposium on Industrial Electronics (ISIE)*, 2022, pp. 1076–1081.

[12] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016. [Online]. Available: https://books.google.de/books?id=LchFDAAAQBAJ

[13] V. Kumar and K. Paul, "Device fingerprinting for cyber-physical systems: A survey," vol. 55, no. 14s, Jul. 2023. [Online]. Available: https://doi.org/10.1145/3584944

[14] R. Colelli, C. Foglietta, R. Fusacchia, S. Panzieri, and F. Pascucci, "Blockchain application in simulated environment for cyber-physical systems security," in *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, 2021, pp. 1–7.

[15] W. Davis, M. Yaqoob, L. Bennett, S. Mihai, D. V. Hung, R. Trestian, M. Karamanoglu, B. Barn, and H. Nguyen, "An innovative blockchain-based traceability framework for industry 4.0 cyber-physical factory," in *Proceedings of the 2023 5th Asia Pacific Information Technology Conference*, ser. APIT '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 118–122. [Online]. Available: https://doi.org/10.1145/3588155.3588174

[16] S. Schorradt, E. Bajramovic, and F. Freiling, "On the feasibility of secure logging for industrial control systems using blockchain," in *Proceedings of the Third Central European Cybersecurity Conference*, ser. CECC 2019. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3360664.3360668

[17] M. K. Choi, C. Y. Yeun, and P. H. Seong, "A novel monitoring system for the data integrity of reactor protection system using blockchain technology," *IEEE Access*, vol. 8, pp. 118 732–118 740, 2020.

[18] S. Otte, L. Reuscher, D. Keller, and J. Fleischer, "Blockchain architecture for process-level traceability of continuous mixing process in battery cell production," in *2024 1st International Conference on Production Technologies and Systems for E-Mobility (EPTS)*, 2024, pp. 1–14.

[19] A. Parvizimosaed, H. Azad, D. Amyot, and J. Mylopoulos, "Protection against ransomware in industrial control systems through decentralization using blockchain," in *2023 20th Annual International Conference on Privacy, Security and Trust (PST)*, 2023, pp. 1–5.

[20] C. T. B. Garrocho, K. N. Oliveira, D. J. Sena, C. F. M. da Cunha Cavalcanti, and R. A. R. Oliveira, "Bace: Blockchain-based access control at the edge for industrial control devices of industry 4.0," in *2021 XI Brazilian Symposium on Computing Systems Engineering (SBESC)*, 2021, pp. 1–8.

[21] Z. Jadidi, A. Dorri, R. Jurdak, and C. Fidge, "Securing manufacturing using blockchain," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, pp. 1920–1925.

[22] S. S. Kirkman, S. Fulton, J. Hemmes, C. Garcia, and J. C. Wilson, "A blockchain architecture to increase the resilience of industrial control systems from the effects of a ransomware attack: A proposal and initial results," *ACM Trans. Cyber-Phys. Syst.*, vol. 8, no. 1, Jan. 2024. [Online]. Available: https://doi.org/10.1145/3637553

[23] J. Hayes, A. Aneiba, and M. Gaber, "Symbiot: Towards an extensible blockchain integration testbed for iiot," in *Proceedings of the 1st Workshop on Enhanced Network Techniques and Technologies for the Industrial IoT to Cloud Continuum*, ser. IIoT-NETs '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 8–14. [Online]. Available: https://doi.org/10.1145/3609389.3610565

[24] M. M. Cook, A. K. Marnerides, and D. Pezaros, "Plcprint: Fingerprinting memory attacks in programmable logic controllers," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3376–3387, 2023.

[25] Q. Gu, D. Formby, S. Ji, H. Cam, and R. Beyah, "Fingerprinting for cyber-physical system security: Device physics matters too," *IEEE Security & Privacy*, vol. 16, no. 5, pp. 49–59, 2018.

[26] Z. Lu, "A robust anomaly detection approach for iiot time series," in *Proceedings of the 2024 2nd International Conference on Frontiers of Intelligent Manufacturing and Automation*, ser. CFIMA '24. New York, NY, USA: Association for Computing Machinery, 2025, p. 168–173. [Online]. Available: https://doi.org/10.1145/3704558.3707091

[27] T. Roy and A. A. L. Beex, "Power measurement based code classification for programmable logic circuits," in *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2018, pp. 644–648.

[28] A. Keliris and M. Maniatakos, "Remote field device fingerprinting using device-specific modbus information," in *2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2016, pp. 1–4.

[29] C. T. B. Garrocho, K. N. Oliveira, A. L. d. Santos, C. F. M. da Cunha Cavalcanti, and R. A. R. Oliveira, "Toward a failures model for communication of decentralized applications with blockchain networks applied in the industrial environment," *IEEE Wireless Communications*, vol. 29, no. 3, pp. 40–46, 2022.

[30] A. S. M. S. Hosen, P. K. Sharma, D. Puthal, I.-H. Ra, and G. H. Cho, "Secblock-iiot: A secure blockchain-enabled edge computing framework for industrial internet of things," in *Proceedings of the Third International Symposium on Advanced Security on Software and Systems*, ser. ASSS '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3591365.3592945

[31] L. Ribeiro and M. Björkman, "Transitioning from standard automation solutions to cyber-physical production systems: An assessment of critical conceptual and technical challenges," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3816–3827, 2018.

[32] OSHA. [Online]. Available: https://www.osha.gov/etools/electric-power/illustrated-glossary/sub-station

[33] K. Stouffer, M. Pease, C. Tang, T. Zimmerman, V. Pillitteri, and S. Lightman, "Guide to operational technology (ot) security," National Institute of Standards and Technology, Tech. Rep., 2022.

[34] M. Memon, S. S. Hussain, U. A. Bajwa, and A. Ikhlas, "Blockchain beyond bitcoin: Blockchain technology challenges and real-world applications," in *2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, 2018, pp. 29–34.

[35] W. Wang and J. Li, "A feasibility study on containerization of traditional embedded systems," in *2024 4th International Conference on Electronic Information Engineering and Computer Communication (EIECC)*, 2024, pp. 217–221.

[36] S. Shibuya, K. Kobayashi, T. Ohkubo, K. Watanabe, K. Tian, N. J. Sebi, and K. C. Cheok, "Seamless rapid prototyping with docker container for mobile robot development," in *2022 61st Annual Conference of the Society of Instrument and Control Engineers (SICE)*, 2022, pp. 1063–1068.

[37] W. Wang, "Research on using docker container technology to realize rapid deployment environment on virtual machine," in *2022 8th Annual International Conference on Network and Information Systems for Computers (ICNISC)*, 2022, pp. 541–544.